Informatik III - Tutorium IX & X (SR -107)

Tut Nr. 9 – Wiederholung, Komplexitätsklassen

David Münch

Universität Karlsruhe (TH)
Institut für Informatik
IAKS Beth

14. Januar 2009

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/~uhbro/

Tutorium 09: Mittwochs 8:00 Uhr - Raum -107
Tutorium 10: Mittwochs 9:45 Uhr - Raum -107

Übungsblattabgabe Donnerstag.

Wichtig!

Die Onlineanmeldung für den Schein und die Klausur ist ab sofort möglich.

Es gibt zum neuen Thema ein vorlesungsbegleitendes Skriptum unter: `https://puck.iaks.uka.de/eiss/fileadmin/User/Info_3/skript.pdf`.

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Wiederholung alter Stoff

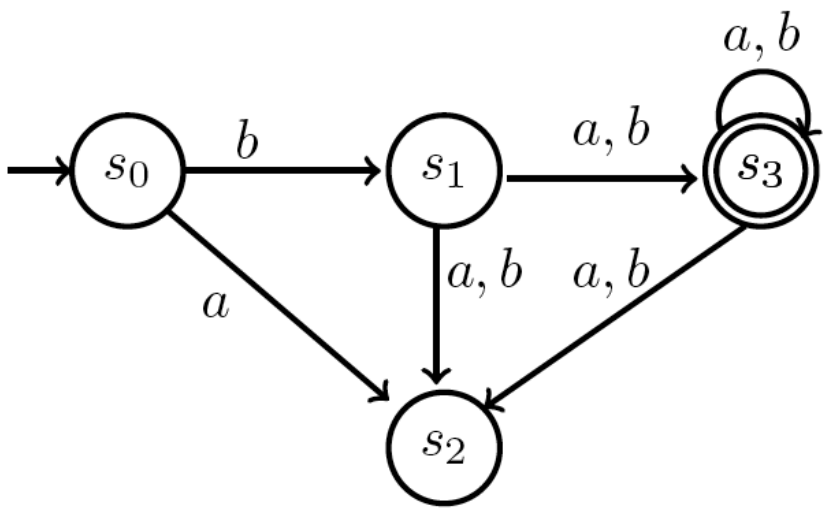
Was wollen wir heute erreichen?

- Wiederholung alter Stoff
- Wiederholung Komplexitätstheorie

Aufgabe

Aus der Klausur 'Informatik III' vom 8.3.2006

Gegeben sei der untenstehende nicht deterministische endliche Akzeptor. Startzustand s_0, Endzustand s_3. Geben Sie einen äquivalenten deterministischen Akzeptor an.



Potenzmengenkonstruktion

Gegeben sei ein NEA $A := (Q, \Sigma, \delta, s, F)$. Wir konstruieren daraus einen DEA $\tilde{A} = (\tilde{Q}, \Sigma, \tilde{\delta}, \tilde{s}, \tilde{F})$

Potenzmengenkonstruktion

Gegeben sei ein NEA $A := (Q, \Sigma, \delta, s, F)$. Wir konstruieren daraus einen DEA $\tilde{A} = (\tilde{Q}, \Sigma, \tilde{\delta}, \tilde{s}, \tilde{F})$

- $\tilde{Q} = 2^Q$, d.h. die Zustände des DEA sind Mengen von Zuständen des NEA.

Potenzmengenkonstruktion

Gegeben sei ein NEA $A := (Q, \Sigma, \delta, s, F)$. Wir konstruieren daraus einen DEA $\tilde{A} = (\tilde{Q}, \Sigma, \tilde{\delta}, \tilde{s}, \tilde{F})$

- $\tilde{Q} = 2^Q$, d.h. die Zustände des DEA sind Mengen von Zuständen des NEA.
- $\tilde{\delta} : \tilde{Q} \times \Sigma \to \tilde{Q}$ # interessante Teil

Potenzmengenkonstruktion

Gegeben sei ein NEA $A := (Q, \Sigma, \delta, s, F)$. Wir konstruieren daraus einen DEA $\tilde{A} = (\tilde{Q}, \Sigma, \tilde{\delta}, \tilde{s}, \tilde{F})$

- $\tilde{Q} = 2^Q$, d.h. die Zustände des DEA sind Mengen von Zuständen des NEA.
- $\tilde{\delta} : \tilde{Q} \times \Sigma \to \tilde{Q}$ # interessante Teil
- $\tilde{s} := E(s)$

Potenzmengenkonstruktion

Gegeben sei ein NEA $A := (Q, \Sigma, \delta, s, F)$. Wir konstruieren daraus einen DEA $\tilde{A} = (\tilde{Q}, \Sigma, \tilde{\delta}, \tilde{s}, \tilde{F})$

- $\tilde{Q} = 2^Q$, d.h. die Zustände des DEA sind Mengen von Zuständen des NEA.
- $\tilde{\delta} : \tilde{Q} \times \Sigma \to \tilde{Q}$ # interessante Teil
- $\tilde{s} := E(s)$
- $\tilde{F} := \{\tilde{q} \in \tilde{Q} | \tilde{q} \cap F \neq \emptyset\}$

Lösung:

Startzustand ist $\{0\}$, Endzustand $\{2,3\}$

	a	b
$\{0\}$	$\{2\}$	$\{1\}$
$\{2\}$	$\emptyset$	$\emptyset$
$\{1\}$	$\{2,3\}$	$\{2,3\}$
$\{2,3\}$	- $\{2,3\}$	- $\{2,3\}$

Aufgabe

Betrachten Sie die Sprache $\mathcal{L} = \{a^{2n}b^{3n} | n \geq 1\}$.

1. Geben Sie die Menge der Produktionen vom Typ 2 an, die $\mathcal{L}$ erzeugt.

Aufgabe

Betrachten Sie die Sprache $\mathcal{L} = \{a^{2n}b^{3n} | n \geq 1\}$.

1. Geben Sie die Menge der Produktionen vom Typ 2 an, die $\mathcal{L}$ erzeugt.
2. Zeigen Sie, dass $\mathcal{L}$ sich nicht durch eine Chomsky Grammatik vom Typ 3 erzeugen lässt.

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.
- Grammatiken, bei denen alle Ableitungsregeln die Form

  $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \backslash \{S\})^+$ und $|u| \leq |v|$, oder $S \to \varepsilon$

  haben, heissen **kontextsensitiv** od. Grammatiken vom **Typ 1**.

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.
- Grammatiken, bei denen alle Ableitungsregeln die Form

  $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \backslash \{S\})^+$ und $|u| \leq |v|$, oder $S \to \varepsilon$

  haben, heissen **kontextsensitiv** od. Grammatiken vom **Typ 1**.
- Grammatiken, bei denen alle Ableitungsregeln die Form $A \to v$ mit $A \in V$ und $v \in (V \cup \Sigma)^*$ haben, heissen **kontextfrei** oder Grammatiken vom **Typ 2**.

Definition: Chomsky-Hierarchie

- Grammatiken ohne weitere Einschränkungen heissen Grammatiken vom **Typ 0**.
- Grammatiken, bei denen alle Ableitungsregeln die Form

  $u \to v$ mit $u \in V^+, v \in ((V \cup \Sigma) \backslash \{S\})^+$ und $|u| \leq |v|$, oder $S \to \varepsilon$

  haben, heissen **kontextsensitiv** od. Grammatiken vom **Typ 1**.
- Grammatiken, bei denen alle Ableitungsregeln die Form $A \to v$ mit $A \in V$ und $v \in (V \cup \Sigma)^*$ haben, heissen **kontextfrei** oder Grammatiken vom **Typ 2**.
- Grammatiken, bei denen alle Ableitungsregeln die Form $A \to v$ mit $A \in V$ und $v = \varepsilon$ oder $v = aB$ mit $a \in \Sigma, B \in V$ haben, heissen **rechtslinear** oder Grammatiken vom **Typ 3**.

Satz: Pumping-Lemma

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n, v \neq \epsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Satz: Pumping-Lemma

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n, v \neq \epsilon,$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Hinweis:
Man verwendet das Pumping-Lemma um zu zeigen, dass eine Sprache nicht regulär ist.

Lösung:

1. $S \rightarrow a^2Sb^3|a^2b^3$

Lösung:

1. $S \to a^2Sb^3|a^2b^3$
2. Da die Sprachen des Typs 3 die regulären Sprachen sind, verwenden wir das Pumplemma.
   Sei $\mathcal{L}$ regulär und k die Zahl gemäss Pumplemma. Da $|a^{2k}b^{3k}| \geq k$. gibt es eine Zerlegung $a^{2k}b^{3k} = uvw$. Da $|uv| \leq k$, gibt es ein l, sodass $uv = a^l$. Für $v = a^j \ (j \geqslant 1)$ fordert das Pumpinglemma, dass $uv^2w = a^{2k+j}b^{3k} \in \mathcal{L}$. Dies verletzt die definierenden Bedingungen für $\mathcal{L}$. Also ist $\mathcal{L}$ nicht regulär.

Aufgabe

1. Geben Sie einen vollständigen, zustandsminimalen Akzeptor an, der genau die Worte der Länge 3 über dem Alphabet $\{a, b\}$ akzeptiert.

Aufgabe

1. Geben Sie einen vollständigen, zustandsminimalen Akzeptor an, der genau die Worte der Länge 3 über dem Alphabet $\{a, b\}$ akzeptiert.
2. Zeigen Sie, dass der von Ihnen in a angegebene Akzeptor minimal ist.

Minimierungsalgorithmus:

1. Entferne alle überflüssigen Zustände.

Minimierungsalgorithmus:

1. Entferne alle überflüssigen Zustände.
2. Stelle eine Tabelle aller Zustandspaare $\{p, q\}$ mit $p \neq q$ auf.

Minimierungsalgorithmus:

1. Entferne alle überflüssigen Zustände.
2. Stelle eine Tabelle aller Zustandspaare $\{p, q\}$ mit $p \neq q$ auf.
3. Markiere alle Paare $\{p, q\}$ mit entweder $p \in F$ oder $q \in F$.

Minimierungsalgorithmus:

1. Entferne alle überflüssigen Zustände.
2. Stelle eine Tabelle aller Zustandspaare $\{p, q\}$ mit $p \neq q$ auf.
3. Markiere alle Paare $\{p, q\}$ mit entweder $p \in F$ oder $q \in F$.
4. Für jedes noch unmarkierte Paar $\{p, q\}$ und jedes $a \in \Sigma$ teste, ob $\{\delta(p, a), \delta(q, a)\}$ bereits markiert ist. Wenn ja, dann markiere auch $\{p, q\}$.

Minimierungsalgorithmus:

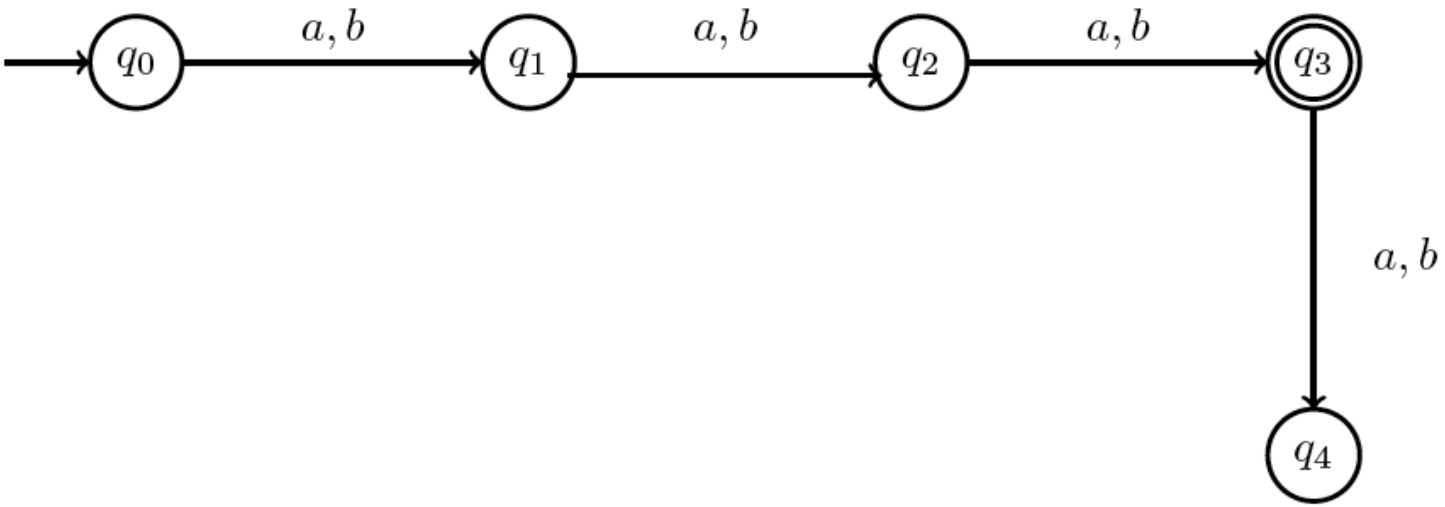
1. Entferne alle überflüssigen Zustände.
2. Stelle eine Tabelle aller Zustandspaare $\{p, q\}$ mit $p \neq q$ auf.
3. Markiere alle Paare $\{p, q\}$ mit entweder $p \in F$ oder $q \in F$.
4. Für jedes noch unmarkierte Paar $\{p, q\}$ und jedes $a \in \Sigma$ teste, ob $\{\delta(p, a), \delta(q, a)\}$ bereits markiert ist. Wenn ja, dann markiere auch $\{p, q\}$.
5. Wiederhole Schritt 4 bis sich keine Änderung in der Tabelle mehr ergibt.

Minimierungsalgorithmus:

1. Entferne alle überflüssigen Zustände.
2. Stelle eine Tabelle aller Zustandspaare $\{p, q\}$ mit $p \neq q$ auf.
3. Markiere alle Paare $\{p, q\}$ mit entweder $p \in F$ oder $q \in F$.
4. Für jedes noch unmarkierte Paar $\{p, q\}$ und jedes $a \in \Sigma$ teste, ob $\{\delta(p, a), \delta(q, a)\}$ bereits markiert ist. Wenn ja, dann markiere auch $\{p, q\}$.
5. Wiederhole Schritt 4 bis sich keine Änderung in der Tabelle mehr ergibt.
6. Verschmelze alle jetzt noch unmarkierten Paare zu jeweils einem Zustand.

Lösung:

Der untenstehende Akzeptor mit vier Zuständen akzeptiert die gesuchte Sprache. Einziger Endzustand ist $q3$. FEHLER: Übergang von q4 nach q4 mit a oder b fehlt in der Grafik.



Lösung:

Wir bestimmen den Äquivalenzklassenautomaten:

1	x_2			
2	x_1	x_1		
3	x_0	x_0	x_0	
4	x_3	x_2	x_1	x_0
	0	1	2	3

Da alle Zustandspaare als nicht äquivalent markiert werden, ist der Automat minimal.

Aufgabe

Die Grammatik $\mathcal{G}_1$ habe die Produktionen

$P_1 = \{S \to aCa, S \to aSa, S \to aa.C \to Cc, C \to c\}$.
und $\{a, c\}$ als Terminalalphabet und S als Startzeichen.

1. Wandeln Sie die Grammatik in Chomsky-Normalform um und überprüfen Sie mit dem Cocke-Younger-Kasami- Algorithmus, ob $aacaa$ und $aacaaa$ aus $\mathcal{G}_1$ erzeugt werden kann und

Aufgabe

Die Grammatik $\mathcal{G}_1$ habe die Produktionen

$P_1 = \{S \to aCa, S \to aSa, S \to aa.C \to Cc, C \to c\}$.
und $\{a, c\}$ als Terminalalphabet und S als Startzeichen.

1. Wandeln Sie die Grammatik in Chomsky-Normalform um und überprüfen Sie mit dem Cocke-Younger-Kasami- Algorithmus, ob $aacaa$ und $aacaaa$ aus $\mathcal{G}_1$ erzeugt werden kann und
2. Geben Sie einen Kellerautomaten an, der die von $\mathcal{G}_1$ erzeugte Sprache akzeptiert.

Lösung:

FALSCH!

ursprüngliche Regel	resultierende Regeln in CNF
$C \to Cc$	$C \to CC_c,\ C_c \to c$
$S \to aa$	$S \to S_aS_a,\ S_a \to a$
$S \to aSa$	$S \to S_aS_2,\ S_2 \to SS_a$
$S \to aCa$	$S \to S_aC_2,\ C_2 \to CS_a,\ C \to c$

Lösung:

FALSCH!

ursprüngliche Regel	resultierende Regeln in CNF
$C \to Cc$	$C \to CC_c,\ C_c \to c$
$S \to aa$	$S \to S_aS_a,\ S_a \to a$
$S \to aSa$	$S \to S_aS_2,\ S_2 \to SS_a$
$S \to aCa$	$S \to S_aC_2,\ C_2 \to CS_a,\ C \to c$

$aacaa \in L(G_1)$

Lösung:

FALSCH!

ursprüngliche Regel	resultierende Regeln in CNF
$C \to Cc$	$C \to CC_c,\ C_c \to c$
$S \to aa$	$S \to S_aS_a,\ S_a \to a$
$S \to aSa$	$S \to S_aS_2,\ S_2 \to SS_a$
$S \to aCa$	$S \to S_aC_2,\ C_2 \to CS_a,\ C \to c$

$aacaa \in L(G_1)$
$aacaaa \notin L(G_1)$

Lösung:

FALSCH!
Der Kellerautomat $M := (\{a, c\}, \{A, S\}, \{q_0, q_1, q_2, q_3\}, \delta, q_0, S)$ mit

$$
\begin{array}{lccl}
\delta\colon & \mathcal{A}_\lambda \times \mathcal{Q} \times \mathcal{K} & \to & \mathcal{P}_\lambda(\mathcal{Q} \times \mathcal{K}^*) \\
& (a, q_0, S) & \mapsto & (q_0, AS) \\
& (a, q_0, A) & \mapsto & (q_0, AA) \\
& (c, q_0, A) & \mapsto & (q_1, A) \\
& (c, q_1, A) & \mapsto & (q_1, A) \\
& (a, q_1, A) & \mapsto & (q_2, \lambda) \\
& (a, q_2, A) & \mapsto & (q_2, \lambda) \\
& (\lambda, q_2, S) & \mapsto & (q_3, \lambda)
\end{array}
$$

Wiederholung DTM, NTM und NTM als Orakelturingmaschine.

Aufgabe

Gegeben ist der folgende Algorithmus, der prüft ob eine Zahl $n \geq 2$ prim ist.

$z := PRIM(n)$
  $z := 1;$
  $i := 2;$
  while $i \neq n$ **do**
    if $n\%i = 0$ **then** $z := 0;$ **end**
    $i := i + 1;$
  end

Gib die Komplexität des Algortihmus in $|n|$ und in der Länge der Binärdarstellung von n an.

Sei M eine TM mit Eingabealphabet A. Die Rechenzeit von M wird durch die Funktion $t_M : A^* \to \mathbb{N}_0$ beschrieben.

$$t_M(\alpha) = \begin{cases} \text{minimale Zahl von Rechenschritten,} & \text{falls } \alpha \in L(M) \\ \text{die } M \text{ braucht, um } \alpha \text{ zu akzeptieren,} & \\ 0, & \text{falls } \alpha \notin L(M). \end{cases}$$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:
DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

NP $= \bigcup_{k=1}^{\infty} NTIME(n^k)$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

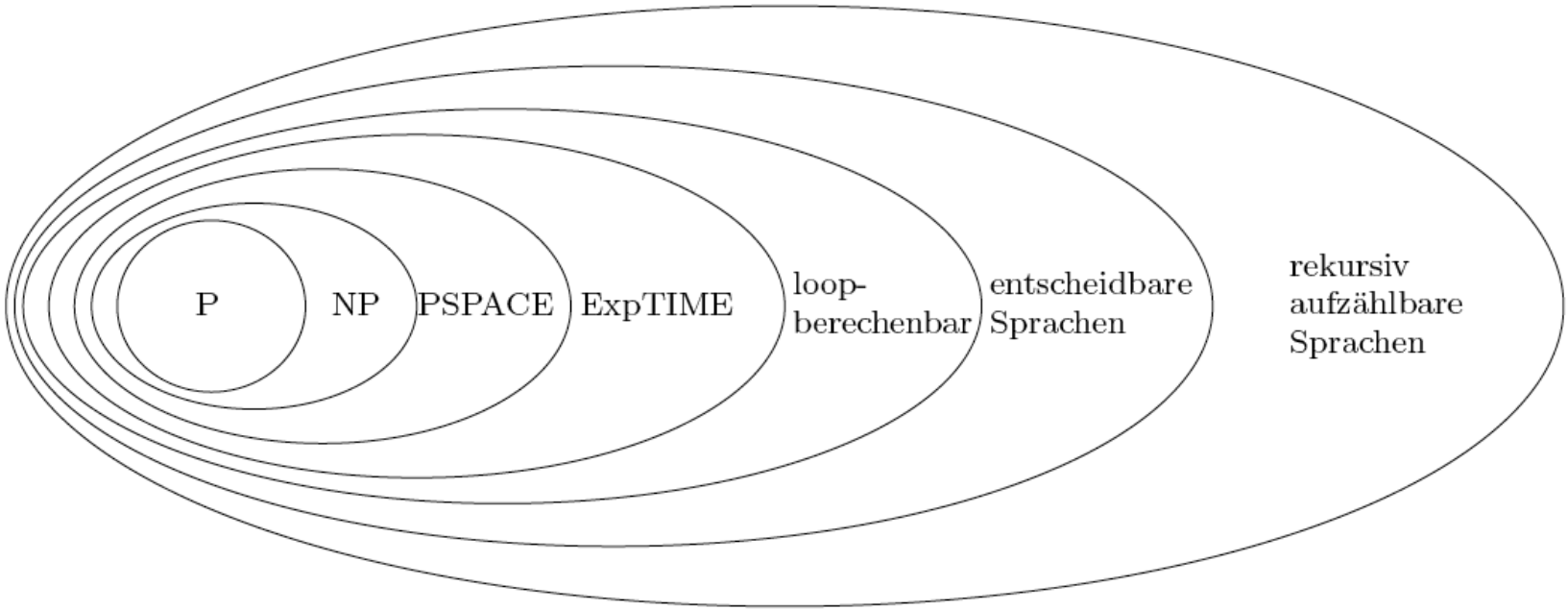
NTIME$(f(n)) =$
$\{L \subset A^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

NP $= \bigcup_{k=1}^{\infty} NTIME(n^k)$

EXPTIME $= \bigcup_{k=1}^{\infty} DTIME(2^n)$

Zentrale Frage: $\mathcal{P} = \mathcal{NP}$?


P
NP
PSPACE
ExpTIME
loop-
berechenbar
entscheidbare
Sprachen
rekursiv
aufzählbare
Sprachen

Aufgabe 1

max2sat sein denke ich

9

Problem 2SAT

- **Gegeben:** Menge U von Variablen, Menge C von Klauseln über U, wobei jede Klausel genau **zwei** Literale enthält.
- **Gefragt:** Existiert eine erfüllende Wahrheitsbelegung für C?

Aufgabe 1

10

2SAT kann in polynomialer Zeit gelöst werden!

Idee: Betrachte eine Klausel $(l_1 \vee l_2)$ mit Literalen l_1 und l_2.

- ist l_1 false, so muss l_2 true sein $(\overline{l_1} \Rightarrow l_2)$.
- ist l_2 false, so muss l_1 true sein $(\overline{l_2} \Rightarrow l_1)$.

Aufgabe 1

11

Zu einer 2SAT-Instanz $(U = \{x_1, \ldots x_n\}, C)$ konstruiere einen Graphen $G = (V, E)$ mit

- $V = \{x_1, \ldots x_n, \overline{x}_1, \ldots \overline{x}_n\}$
- $E = \{(\overline{l_1}, l_2) \mid (l_1 \vee l_2) \text{ oder } (l_2 \vee l_1) \in C\}$

Erklärung: Es werden für jede Klausel $(l_1 \vee l_2)$ die Kanten $\overline{l_1} \rightarrow l_2$ und $\overline{l_2} \rightarrow l_1$ zum Graphen zugefügt.

Die Transformation ist polynomial.

Aufgabe 1

12

Genau wenn es in G Knoten x_i und $\overline{x}_i$ gibt, für die ein gerichteter Weg von x_i nach $\overline{x}_i$ und umgekehrt existiert, dann ist die 2SAT-Instanz nicht lösbar.

Beispiel:

$$(x \vee y) \wedge (\overline{y} \vee x) \wedge (\overline{x} \vee z) \wedge (\overline{z} \vee \overline{x})$$

Aufgabe 1

13

$\Longrightarrow$

Angenommen es gibt Knoten x_i und $\overline{x}_i$ gibt, für die ein gerichteter Weg von x_i nach $\overline{x}_i$ und umgekehrt existiert.

wegen Weg von x_i nach $\overline{x_i}$

$$x_i \Rightarrow \overline{x}_i$$

wegen Weg von $\overline{x_i}$ nach x_i

$$\overline{x}_i \Rightarrow x_i$$

Widerspruch zur Lösbarkeit!

Aufgabe 1

14

$\Longleftarrow$

Eine erfüllende Belegung kann wie folgt gefunden werden:

Setze

$x_i = \mathit{false}$, falls für das Paar $x_i, \overline{x_i}$, ein Pfad von $\overline{x_i}$ nach x_i existiert
$x_i = \mathit{true}$, sonst

Das ist in Polynomialzeit durchführbar. Beweis der Korrektheit durch Induktion.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- nichts Neues, nur Altes wiederholt.

Noch Fragen?

Vorschau

Vorschau

- weitere Komplexitätsklassen

Vorschau

- weitere Komplexitätsklassen
- ???

Bis zum nächsten Mal